* **Client and Cluster mode :**

Client mode : Driver runs on edge node/client machine.

Cluster mode: Driver runs on the cluster.

In PROD environment, we will go with cluster mode because in client mode as soon as you turn off client machine, the application will stop.

* **What is partition skew, what are reasons for it, How to solve partition skew issues?**

Initially whatever partitions we get are generally evenly sized . Partition skew will happen when after doing with wide transformation like join, groupBy so that same key goes in the same partition. So, some partitions has less data, some partitions has more data . Partition skew happens mostly after performing a wide transformation and when we are partitioning different sizes.

Summary: So initially whatever spark creates partitions that are roughly of same size but after wide transformation , some partitions can be over filled or less filled based on the data. This can lead to performance issues. If we have 1 partition with 1GB data and another partition with 20MB data , then one of the task has to do more work and it will result in a degraded performance and it can even lead to OOM issues and errors. These problems can be solved by making sure partitions are of even size.

In order to solve partition skew issues, we can use AQE. If we know data better, we can use salting technique. Salting means if key is occurring more then we try to add random number like 1,2,3 ... we are splitting into many different keys.

If key is coming 10 million times and we add a 10 random numbers to it, then we can get 10 different keys out of it and one would be coming 1 lakh times.

* **What is broadcast join in apache spark**

Any table smaller than 10MB will be broadcasted.

Whenever we have one smaller table and one bigger table, we can implement broadcast join.

If the smaller table or small data frame is broadcasted across all executors, there won’t be shuffling across bigger data. In this way , performing join with out any shuffling and this saves a lot of time .

* **What is difference between partitioning and bucketing?**

Partitioning and bucketing are strategies to structure our data so it will end up scanning part of data and skip some of it. Partitioning works by dividing the data logically. Let’s say we have less no of distinct col values , then we can use partitioning. If we have col name as state with 20 different states then it will result in 20 different folders. When we write query as select \* from table where state = “ “. Then it just scans that particular folder . so instead of scanning all 20 folders we will just scan 1 folder.

Let’s say we have customer table and has millions of customers ,we can’t create partition on customer\_ID. Based on MOD or HASH function , it will create those many files (depending on no of buckets).

Partitioning is folders created on a logic. Bucketing is files created based on hash function.

When we use filter and all in query, partitioning and bucketing helps to get the result much faster.

Bucketing helps for join optimization when we are joining 2 large tables.

* **What are different types of joins in spark or how to do join optimizations?**

Broadcast hash join

Shuffle hash join

Shuffle Sort merge join

Broadcast hash join : When we have one small table and one bigger table, then we will implement this join . there won’t be shuffling involved.

Shuffle sort merge join 🡪 implemented on 2 larger tables. First it will shuffle the data . same keys from both the table goes to same executor, then data will be sorted and then will be merged. This is how normal join works.

Shuffle hash join is slight optimization on shuffle sort merge join. Where we have 1 medium table and 1 large table , we can implement this which means almost everything happens the same way that both tables are shuffled so that same keys . in 1 medium sized table, hash table is created so that there is no need of doing sorting. Sorting itself is a costly operation.

* **Why count when used with group by is transformation else it is an action?**

Df.count() 🡪 action , bcoz simply we are getting some number, we cant perform again some actions on it

Df.groupBy.count() 🡪 transformation because lets say we got 1 lakh keys after group by. Later on we can do filter and do some actions or other transformations so that’s why its an transformation.

* **If our spark job is running slow, how would you approach to debug it / what all performance techniques they have used /how do you optimize performance of our job.**

Follow the below steps:

* Will check spark UI for our slow tasks. We can go and see which task is running slow. If any of the task running slow, which means there is partition skew and we can enable AQE. If we are on spark 3 or .3 versions, aqe will be enabled by default or we can enable AQE for handling partition skew. We can use salting if we are on previous versions of spark.
* Sometimes join may take a lot of time,we can optimize join strategies, lets say in logic code, shuffle sort merge join is happening. If one table is medium sized, then we can invoke shuffle hash join instead of shuffle sort merge join ,so there by sorting is avoided . broadcast hash join > shuffle hash join > shuffle sort merge join . performance fast order.
* Lets say we haven’t given enough resources for our job,lets say we have 1000 partitions but we have 5 CPU cores , iterations to complete that work is more. So ensure sufficient resources are allocated. If we give more resources, then we can do more things in parallel.
* Verify no of data frame partitions. Initial no of partitions are generally controlled by system, but after that we performed wide transformation , by default we have 200 partitions and we have 1000 CPU cores, and 800 will sit idle. So we can change no of shuffle partitions if required.
* If we have more of JVM memory,it subject to Garbage collector which takes a lot of time, so we can mitigate GC delays by giving some off heap memory. So we can give combination of JVM MEMORY AND off heap memory so that GC delays are not taking so much time.
* If we wont have enough memory per CPU core, then while executing the task , it might spill the data to disk. Monitor disk spills, allocate more memory per CPU core if needed.
* If in query sort aggregate is happening, then we can use hash aggregate by changing the query so sorting will be avoided. Opt for hash aggregation over sort aggregation when aggregating. If there are too many distinct values, then it’s not possible.
* Implement caching
* Choose the right file formats and compression techniques .
* **Difference between Managed and External Table? When do you go about creating external tables?**

Any table has data and metadata.

Managed Table: Data and Metadata is managed by spark .

External Table: Data is kept externally and Metadata is managed in spark.

When we drop managed table, both data and metadata gets dropped. Everything will be managed by us in default. Where as in external table, only metadata only will be dropped bcoz its managed by us and data is kept externally, someone else might also be using it so data wont be dropped here in external table.

External Table best suited when same data set multiple teams are using and no one should have permission to drop or make changes to that.

* **Why we are not using mapreduce these days, What are similarities between mapReduce and spark?**

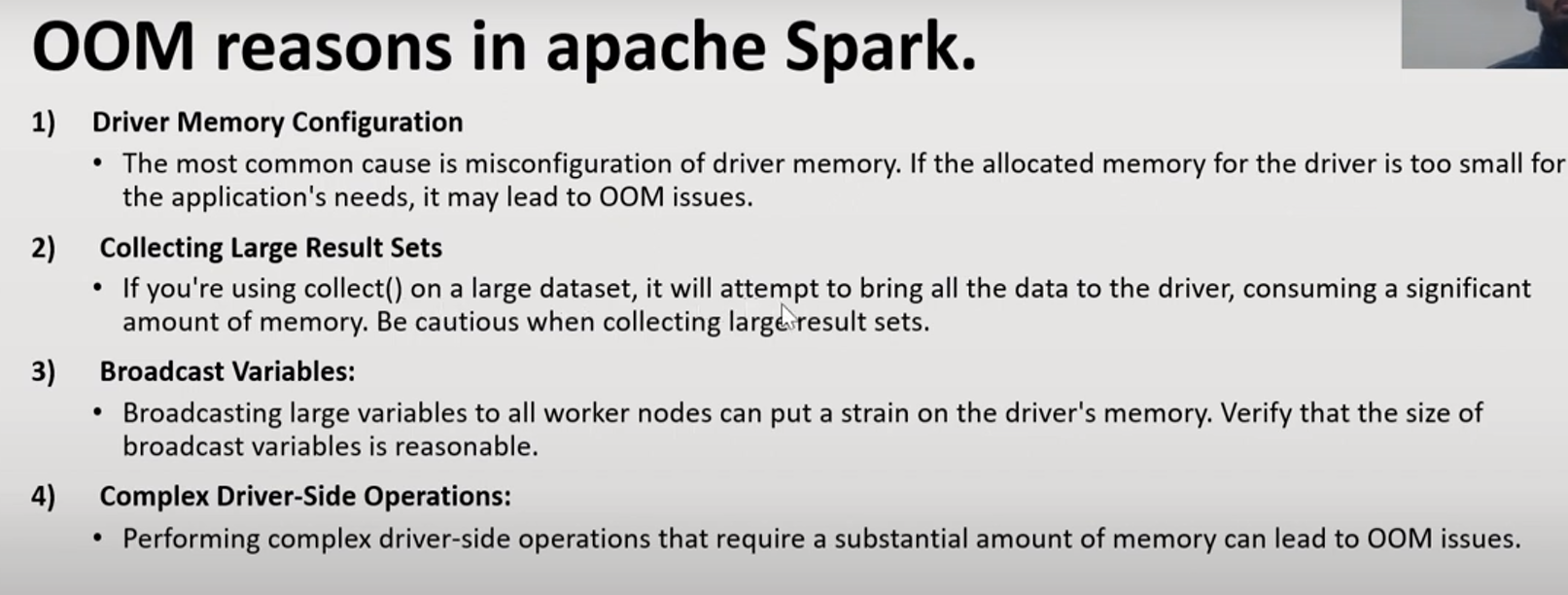
mapReduce will writes data to disc.It uses disc heavily. mapReduce code can only be written in java which is little hard to write. For simple thing we have to write bunch of code.

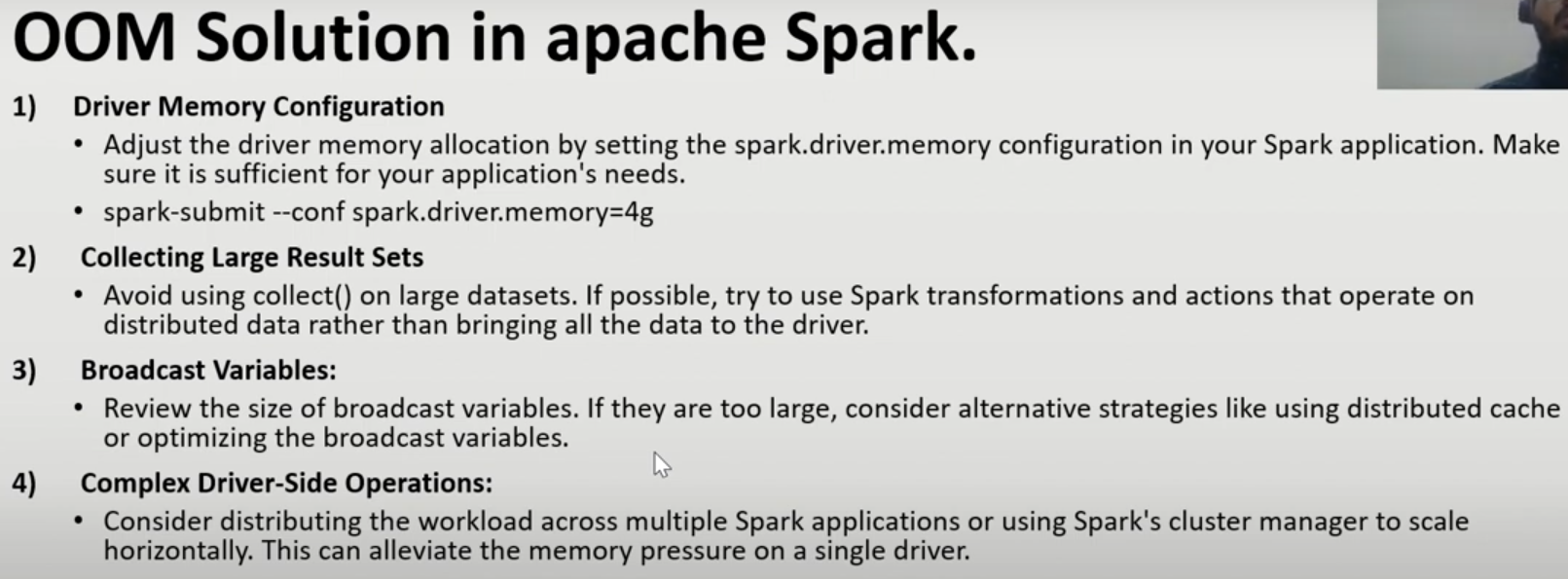
Both mapReduce and spark are for distributed processing/computing. Spark is much faster that mapReduce because spark uses in memory processing.

* **CI-CD Process**
* **OOM issue in Apache Spark?**

OOM issue in spark will occur both in driver and as well as executors. Both are of different reasons.

Driver-OOM Reasons:





**Difference between ADLS Gen1 and ADLS Gen2 and Blob Storage.**

In ADLS Gen1, data is in the form of hierarchical namespace files and folder support, where as in ADLS Gen2 , hierarchical namespace is in the form of containers, files and folders. In Blob, there is container, inside containers it has data in the form of BLOB’s.

ADLS Gen1 supports LRS and GRS where as ADLS Gen2 supports LRS, GRS,ZRS and GZRS whereas blob supports LRS, GRS and ZRS.

ADLS Gen1 supports authentication of AAD, MI, SP, in ADLS Gen2 it supports AAD, MI,SP and SAS. In blob, it supports Access keys and SAS.